Suitability of Addition-Composition Fully Homomorphic Encryption Scheme for Securing Data in Cloud Computing

Richard Omollo[1] and George Raburu[2]

[1,2]Department of Computer Science and Software Engineering

Jaramogi Oginga Odinga University of Science and Technology

P.O. Box 210-40601, Bondo-Kenya.

Email: comolor@hotmail.com[1] and graburu@hotmail.com[2]

Abstract: Cloud computing is a technological paradigm that enables the consumer to enjoy the benefits of computing services and applications without necessarily worrying about the investment and maintenance costs. This paper focuses on the applicability of a new fully homomorphic encryption scheme (FHE) in solving data security in cloud computing. Different types of existing homomorphic encryption schemes, including both partial and fully homomorphic encryption schemes are reviewed. The study was aimed at constructing a fully homomorphic encryption scheme that lessens the computational strain on the computing assets as compared to Gentry's contribution on partial homomorphic encryption schemes where he constructed homomorphic encryption based on ideal lattices using both additive and multiplicative Homomorphisms. In this study both addition and composition operations implementing a fully homomorphic encryption scheme that secures data within cloud computing is used. The work is founded on mathematical theory that is translated into an algorithm implementable in JAVA. The work was tested by a single computing hardware to ascertain its suitability. The newly developed FHE scheme posted better results that confirmed its suitability for data security in cloud computing.

Keywords: Cloud Computing, Data Security, Fully Homomorphic Encryption Schemes, Addition-Composition

I. INTRODUCTION

Cloud computing paradigm traces back to 1960s suggestion by Professor John McCarthy who proposed the concept of *utility computing.* Utility computing is about computer time-sharing technology leading to a future where computing power and even specific applications provided through utility-type business model [1]. This has been the motivation behind cloud computing, a compacted term within technology circles, which adopts a distributed computing style of integrating web services and data centers [2]. However, this paradigm can be easily extended to grid computing, which is a form of distributed computing that implements a virtual supercomputer composed of a cluster of network or internetworked computing dedicated towards a demanding task.

Cloud computing therefore is an internet based system that provides the technology of outsourcing computational needs away from the data owners. It provides a way of storing and accessing cloud data from anywhere by just connecting to the cloud application using the internet [3]. The cloud users have the choice of settling on cloud services of their interest so as to store and transact their data in a remote data server [4]. This data can be accessed and managed through cloud services provided by the cloud providers chosen by the user.

Despite the flexibility and convenience that comes with cloud computing, the security of cloud data has been one of the greatest challenge to its deployment [5]. In order to address this, cryptography is one approach that guarantees safety of cloud resources [6] since it has the capability of hiding data from unauthorized access. There has been a proposed solution to address issues of data security on cloud computing with the adoption of homomorphic encryption scheme supporting only limited mathematical operations.

Fig. 1 below is illustrates an implementation of an ideal cloud service that supports homomorphic encryption in the clouds.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

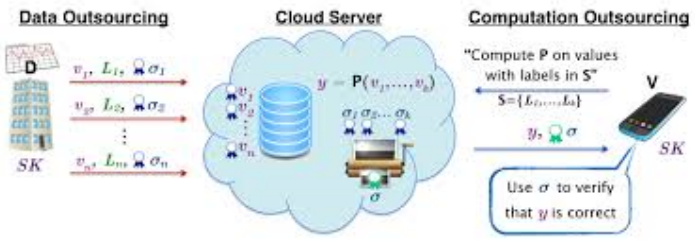


Fig. 1 Homomorphic Encryption in the Clouds

In order to strengthen homomorphic encryptions in clouds a Fully Homomorphic Encryption Scheme (FHE) has been developed that involves more than one operation as proposed by Craig Gentry [7]. More contributions have been made on Gentry's work but still far off from ensuring effective practical applications. As much as FHE schemes have proved versatile, the computing resource demand due to increasing ciphertext size and slowed encryption speed still compromises the implementation. Based on this critical need a new fully homomorphic scheme that is still versatile but also faster in encryption or decryption and guarantees ciphertext size of higher value was developed. This new scheme addresses the existing gap where previously developed schemes strain computing resources such as storage and bandwidth consumption. The use of both additive and composition mathematical operations is adopted in this study to address data security in cloud computing.

II. LITERATURE REVIEW AND RELATED STUDY

2.1 CLOUD COMPUTING

A more functional understanding of Cloud Computing is tied to the concept of the term CLOUD. It [8] derived five logical tautology and concludes that a cloud structure is a six-tuple (space, time, directed graph, set of states, transition function, and initial state) that satisfies the five axioms namely; Common, Location-independent, Online, Utility, and on-Demand (C.L.O.U.D). Cloud computing can therefore be understood to have five distinct essential characteristics: On-Demand Self-Service, Broad Network Access, Resource Pooling, Rapid Elasticity, and Measured Service [9]. Likewise from a hardware point of view, there are three new aspects in the paradigm of cloud computing [10].

2.2 ARCHITECTURE OF CLOUD COMPUTING

The design of Cloud computing architecture encompasses the front-end (a visible interface that cloud users encounter through web-enabled client devices) and the back-end (comprising resources required to deliver cloud computing services). The back-end consists of the bare metal servers, data storage compartments, virtual machines, implemented security mechanism, and cloud service(s) with conformity with a particular deployment model. It is the primary responsibility of the back-end to integrate built-in security mechanisms, traffic management control, and protocols. In the bare metal has the operating system referred to as hypervisor which makes use of well-defined protocols enabling efficient concurrent access onto the virtual machines.

The cloud computing architecture functions in such a way that all applications are controlled and served by a central cloud server with replicated data remotely in the cloud configuration to create limitless efficiency and possibilities.

Fig. 2 below shows a diagrammatic summary of cloud computing architecture.

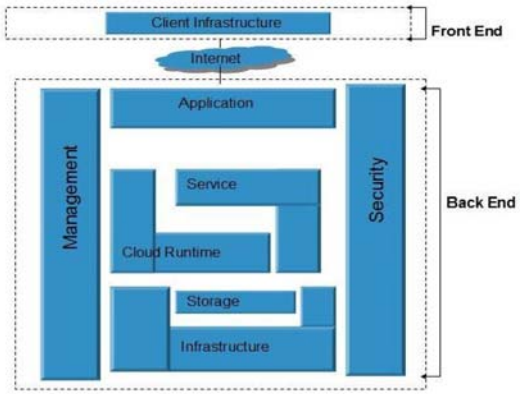


Fig. 2 Cloud Computing Structure

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Cloud services are based on five principal characteristics that demonstrate their relation to, and differences from, traditional computing approaches [11]. These characteristics are summarized in Table 1:

Table 1 Cloud Principal Characteristics

Characteristics	Descriptions
Abstraction of Infrastructure	The computation, network and storage infrastructure resources are abstracted from the application and information resources as a function of service delivery. This abstraction is generally provided by means of high levels of virtualization at the chipset and operating system levels or enabled at the higher levels by heavily customized file systems, operating systems or communication protocols.
Resource Democratization	The infrastructure, applications, or information provides the capability for pooled resources to be made available and accessible to anyone or anything authorized to utilize them using standardized methods
Service Oriented Architecture	Utilization of cloud components in whole or part, alone or with integration, provides a services oriented architecture where resources may be accessed and utilized in a standard way.
Elasticity or Dynamism	The on-demand model of cloud provisioning coupled with high levels of automation, virtualization, and ubiquitous, reliable and high-speed connectivity provides for the capability to rapidly expand or contract resource allocation to service definition and requirements using a self-service model that scales to as-needed capacity.
Utility Model of Consumption and Allocation.	The abstracted, democratized, service-oriented and elastic nature of cloud combined with tight automation, orchestration, provisioning and self-service allows for dynamic allocation of resources based on any number of governing input parameters. Given the visibility at an atomic level, the consumption of resources can then be used to provide a metered utility cost and usage model.

2.3 CLOUD COMPUTING SERVICE DELIVERY MODELS

In the delivery models in cloud computing, there are three service models that commonly established and formalized: Software as a Service (SAAS), Platform as a Service (PAAS) and Infrastructure as a Service (IAAS). The three archetypal models and the derivative combinations thereof generally describe cloud computing service delivery. The three individual models are often referred to as the "SPI MODEL" [11]. There are other specialized variations of the three main service delivery models based on distinct combination of IT resources.

2.3.1 Software as a Service (SaaS): This is a service delivery model where applications are available to the cloud users over the network. The cloud providers take the responsibility of hosting these applications used by the consumers under software distribution model. The model is characterized by application delivery from 1:M model, that is, single-instance, multitenant architecture as opposed to traditional 1:1 model, and also associated with pay-as-you-go subscription licensing model. The distinguishing bit of SaaS from other service delivery model is that it was designed to work with web browsers. The main advantages with this model include; automated updates and patch management services, data compatibility across the enterprise and global accessibility of software of interest by the cloud users. The design is made in such a way that the cloud consumer depends on the service provider in the implementation of security e.g. the provider has to ensure that each cloud user do not access each other's private data [12]. The cloud consumer substitute new software applications with the old one focusing on enhancing the security functionalities provided by legacy application and achieving a successful data migration [13].

2.3.2 Platform as a Service (PaaS): This is service delivery model where all facilities required the support of the complete life cycle of building and delivery of web applications and services entirely available on the Internet. It is an outgrowth of SaaS model but uses web-based development unlike with SaaS where specific operating systems instance is favoured. It encapsulate a layer of software and provide it as a service used to build higher levels services with two perspective of the producer or consumer of services: producing a platform by integrating an operating system, and an encapsulated service presented to them through Application Programming Interface (API) [14]. The advantage with this model is that it provides the entire infrastructure needed by allowing users to focus on innovation and not the complex infrastructure. A secure PaaS cloud can only be achieved through thoroughly inspecting the conceptual security solutions, architectural and software design, implementation and service provisioning [15]. Also for PaaS model, applications are deployed without the necessity of purchasing and maintaining the hardware and software thereby depending on a secure browser. PaaS application security includes the security of application deployed on PaaS as well as the PaaS platform security itself and it is therefore the responsibility of the PaaS provider to protect the runtime engine which runs the client applications [16]. This deployment model has its products available with different development stacks.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

2.3.3 Infrastructure as a Service (IaaS): This is service delivery model where cloud service providers manage the transition and hosting of selected applications on their infrastructure. In this model, the computing services are provided on a virtualized environment with cloud users maintaining ownership and management of the applications while hosting operations and infrastructure management are off-loaded to the cloud provider. The benefits that comes with this service delivery model includes availability of resources on demand from any location 24x7 when required by the cloud users. The physical security of cloud users' data and any chance of system failure is smoothly handled by the cloud provider. In IaaS, the security varies on deployment model adopted in its implementation. For instance, in private cloud there is control over solutions from top to bottom while in public cloud, there is control on the VMS and services running on the VMS.

The three main service delivery models can be captured in the Fig. 3.

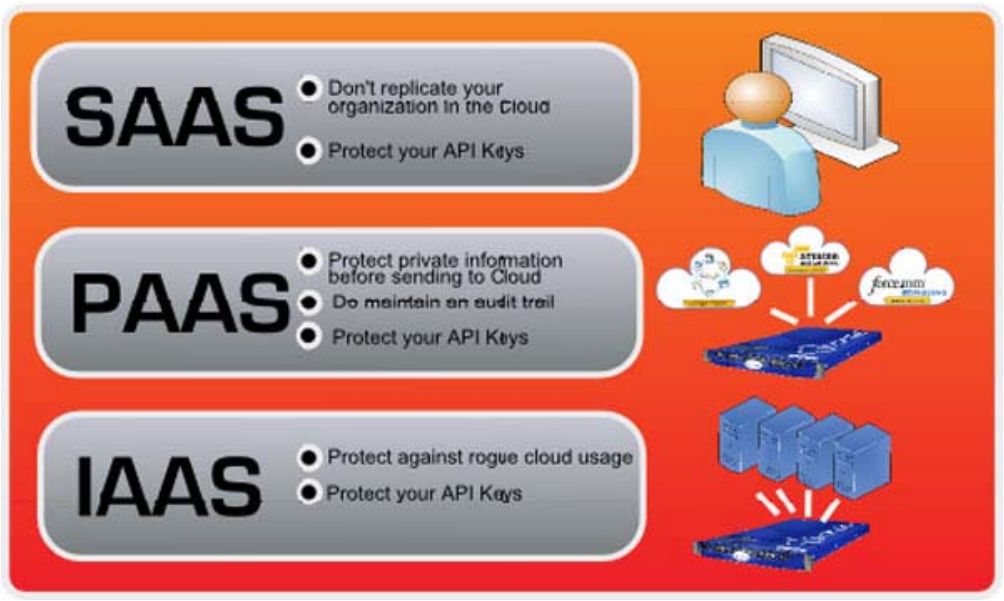


Fig. 3 Security Checklist for SAAS, PAAS, and IAAS

2.4 CLOUD COMPUTING DEPLOYMENT AND CONSUMPTION MODELS

Regardless of the cloud service delivery model utilized, there are five models that can be used to deploy cloud services [11]. For specific organization, cloud integrators play a vital role in determining the right cloud path to be adopted. The models are as listed below:

2.4.1 Private Cloud: Is provisioned exclusive use by a single organization with multi-tenants resource access. It may be owned, managed and operated by the organization, a third party or some combination of them.

2.4.2 Public Cloud: Is provisioned for open use of resources dynamically by the general public. It may be owned, managed and operated by one or more organizations who sells the cloud services.

2.4.3 Community Cloud: Is provisioned for exclusive use by a specific community of clients from organizations that share common interests like mission, security requirements, policy or compliance considerations. It is owned, managed and operated by one or more organization in a community.

2.4.4 Hybrid Cloud: Is composed of two or more distinct cloud infrastructures characteristics (private, public or community) that remain unique entities, but are bounded together by standardized or proprietary technology that enables data and application portability. It is owned, managed and operated by both cloud infrastructures incorporated.

2.4.5 Virtual Private Cloud: Is allocated within a public cloud environment provisioned for a certain level of isolation between the different organizations sharing the cloud resources. It is viewed as a hybrid model of cloud infrastructure in which a private cloud solution is provided within a public cloud infrastructure. Its ownership, management and operations is assumed by the public cloud infrastructure. The Table 2 below summarizes cloud deployment models, with each viewed on access security and infrastructure where it is hosted.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Table 2 Summary of Cloud Deployment Models

Model	Infrastructure		Access	
	Location	Ownership	Trusted	Untrusted
Private	On Premise	Organization	Yes	No
Public	Off Premise	Third Party	No	Yes
Community	Off premise	Third Party	Yes	No
Hybrid	On and Off Premise	Organization and Third Party	Yes	Yes
Virtual Private	Off Premise	Third Party	Yes	Yes

The security issues of the five cloud computing deployment are illustrated in Table 3.

Table 3 Cloud Deployment Models and their Security Issues

Model	Security Issues	Control Issues
Private	Most secure	Most control
Public	Least secure, multi-tenancy, and transfer over the Internet	Least control
Community	Least Secure	Least Control
Hybrid	Control of security between private and public clouds	Least control
Virtual Private	Most secure	Most Control

2.5 CLOUD COMPUTING SECURITY

In cloud architecture the security components and services must be transparent and generic: transparent since there is need for automatic application without much human intervention, and generic to ensure adjustability on the part of clients, requirements, applications and required services. A functional cloud computing security architecture is composed of Security Access Points that provides front-end security services, Security infrastructure servers that manage all cloud stored data registered in the cloud, and secure client for cloud standard clients stations extended with some security components.

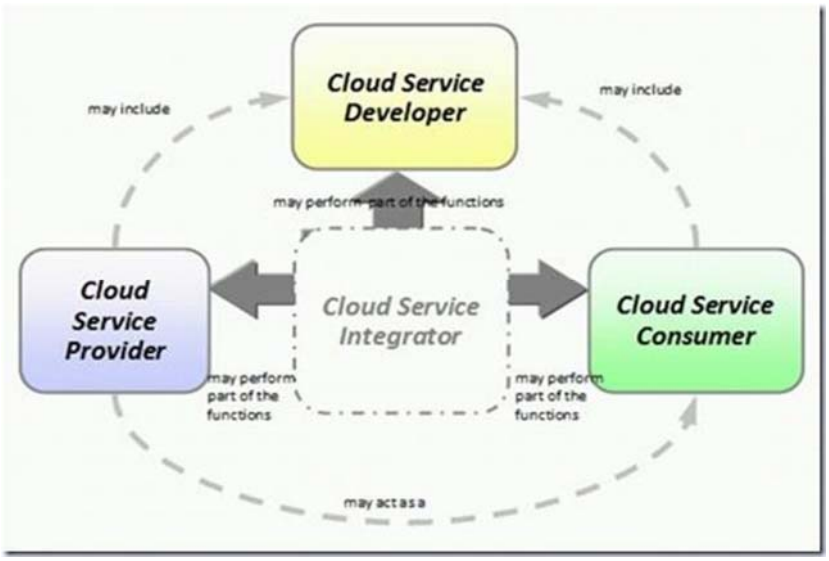
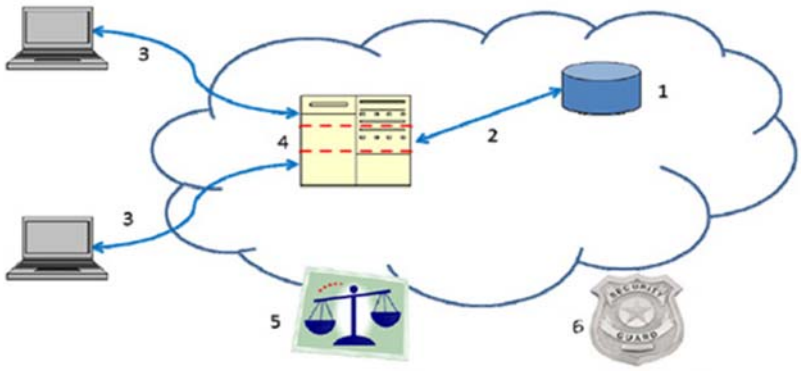


Fig. 4 Cloud Computing Security Infrastructure

The multi-tenancy model and the pooled computing resources has introduced new security challenges such as shared resources on the same physical machines inviting unexpected side channels between machine resources and a regular resource [17]. Security being an important component of cloud computing deployment, it is necessary to address it to enable cloud consumers enjoy its benefits uninterrupted [14].

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Cloud computing architecture has numerous security issues as it encompasses many technologies including networks, databases, operating systems, virtualization, resource scheduling, transaction management, load balancing, concurrency control and memory management. Therefore this means that security issues affecting many of these systems and technologies are applicable to cloud computing. Take for instance the network that interconnects the systems in a cloud has to be secure. Furthermore, virtualization paradigm in cloud computing leads to several security concerns e.g. mapping the virtual machines to the physical machines has to be carried out securely. Data security involves encrypting the data as well as ensuring that appropriate policies are enforced for data sharing. In addition, resource allocation and memory management algorithms have to be secure. Finally, data mining techniques may be applicable for malware detection in the clouds – an approach which is usually adopted in intrusion detection systems (IDSs) ([18], [19], [20], [21], and [22]). Fig. 5 illustrates the six specific areas of the cloud computing environment where equipment and software require substantial security attention [23].



(1) security of data at rest, (2) security of data in transit, (3) authentication of users or applications or processes, (4) robust separation between data belonging to different customers, (5) cloud legal and regulatory issues, and (6) incident response.

Fig. 5 Areas of Security Concerns in Cloud Computing

For securing data at rest, cryptographic encryption mechanisms are certainly the best options. The hard drive manufacturers are now shipping self-encrypting drives that implement trusted storage standards of the trusted computing group [23]. These self-encrypting drives build encryption hardware into the drive, providing automated encryption with minimal cost or performance impact. Although software encryption can also be used for protecting data, it makes the process slower and less secure since it may be possible for an adversary to steal the encryption key from the machine without being detected. Encryption is the best option for securing data in transit as well. In addition, authentication and integrity protection mechanisms ensure that data only goes where the customer wants it to go and it is not modified in transit. Strong authentication is a mandatory requirement for any cloud deployment. User authentication is the primary basis for access control. In the cloud environment, authentication and access control are more important than ever since the cloud and all of its data are accessible to anyone over the Internet. The trusted computing group's (TCG's) IF-MAP standard allows for real-time communication between a cloud service provider and the customer about authorized users and other security issues. When a user's access privilege is revoked or reassigned, the customer's identity management system can notify the cloud provider in real-time so that the user's cloud access can be modified or revoked within a very short span of time [24]. The problem is that the cloud service being a web application, it contains data remembrance or persistence remains an issue due to the replication and distribution of data even after user left a cloud provider. This can be corrected by use homomorphic encryption schemes to protect data on cloud [25].

2.6 CHALLENGES AND SOLUTIONS TO DATA SECURITY IN CLOUD COMPUTING

The security of Cloud Computing is the major concern to be addressed since if the security measures are not provided properly for data operations and transmissions then data is at high risk [26]. This is accelerated by the fact that cloud computing do provide facilities for a group of cloud users to access the stored data thus there is a possibility of having high data risk. Therefore it is necessary to say that strongest security measures need to be implemented by identifying security challenges and solutions to handle these challenges [27].

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

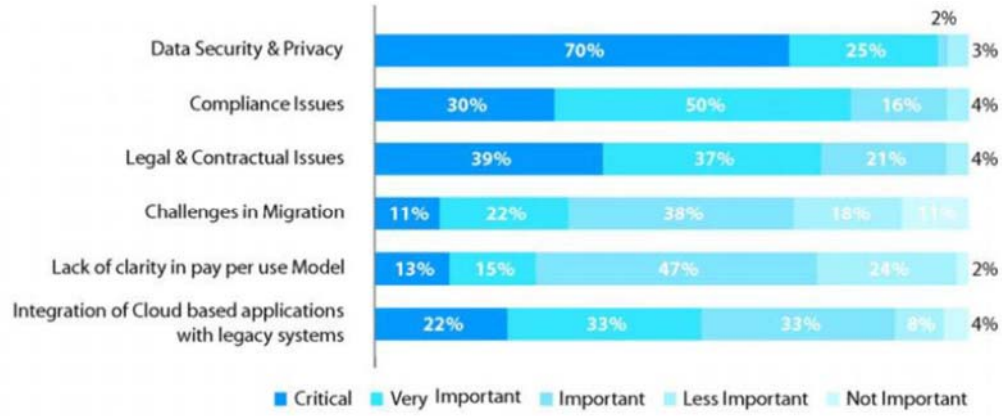


Fig. 6 Factors Influencing Cloud Computing Adoption [27]

From Fig. 6 illustrates the importance of Data Security and Privacy are most important in the adoption of cloud computing.

2.7 HOMOMORPHIC ENCRYPTION SCHEMES

The term *homomorphism* is borrowed from algebra meaning a structure-preserving map between two algebraic structures of the same type. The concept of homomorphism has been employed in explaining *homomorphic*, .which means in literal sense same effect for different expression. Homomorphic encryption enables computation to be performed on the encrypted data without requiring the secret key. This makes it a better approach to enhance security of sensitive data stored and manipulated on untrusted storage systems [28]. Therefore an efficient and fully homomorphic cryptosystem have great practical benefits to outsourcing private computations.

The concept of homomorphic encryption was introduced in 1978 by Rivest, Adleman and Dertouzous after the development of RSA algorithm [29], where they highlighted four possible encryption functions which include RSA as additive and multiplicative privacy homomorphism. Since then, many encryption schemes have been developed but they were either using addition or multiplication homomorphic computations. Later, an encryption scheme was developed [30] that perform unlimited number of addition operations but single multiplication.

The realization of fully homomorphic encryption scheme appeared elusive for more than thirty years until in 2009 when Craig Gentry from IBM implemented the first version that could perform many additions and multiplications using ideal lattices and bootstrapping technique [7]. He based his construction on ideal lattices starting with constructing somewhat homomorphic encryption scheme, limited to evaluating low-degree polynomials over encrypted data. He then modified it to make it bootstrappable hence capable of evaluating its own decryption circuit with at least one more operation. Even though Gentry's scheme was able to demonstrate the possibility of fully homomorphic encryption scheme, there was still need to improve on its complexity, efficiency and performance. For instance, he estimated that building a circuit to execute an encrypted Google search with encrypted keywords would multiply the current computing time by one trillion. More research have been done with the motivation to improve on Gentry's work. These are referred to as second generation of homomorphic encryption schemes and have their security based on the hardness of the Learning with errors problem apart from the LTV scheme [31] whose security is based on a variant of the NTRU computational problem.

2.8 NOISE GROWTH, ATTACKS AND EFFICIENCY IN FULLY HOMOMORPHIC ENCRYPTION SCHEMES

In the context of homomorphic encryption schemes, *noise* is often referred to as a small term added into the ciphertext while encrypting. It depends with the application. It may be a small integer if the scheme is based on integers or a small polynomial if the scheme is based on polynomial. This noise plays a significant role in the encryption process in that it is responsible for the security of the cryptosystem. However, when it comes to decipherment in the same case, the decryption function gets affected by the noise term especially if it is greater than a certain maximum value. Homomorphic operation usually increases the noise and it is advisable to limit the number of operations in order to control it. For instance in the case of fully homomorphic encryption schemes over integers [32], to encrypt a bit m into an integer c with p being the private key and sample values of q and r are considered using the equation:

$$c = pq + 2r + m \quad \text{……..Eqn. 1}$$

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

For decrypting the value *c*, *c mod p* gives *2r+m* only if *2r+m* is smaller than *p* thus *mod 2* is reduced to end up with *m*. Also it is important to note that the decryption won't work if *2r* is bigger than *p*, thus makes *2r* to be the noise term in this case. This is explained further when the homomorphic property of the scheme is considered. For example in a case where there are two ciphertexts c_1 and c_2, then applying the addition operation will end up with $c_1 + c_2 = p(q_1 + q_2) + 2(r_1 + r_2) + (m_1 + m_2)$ hence increasing the noise. The multiplication operation further complicates it and therefore with each operation increasing the noise, it is advisable to set an acceptable maximum value for the noise by limiting the number of operations.

The significance of noise term in the construction of fully homomorphic encryption scheme is an important factor to be considered. Practical constructions should consider the reduction of noise term as it complicates its decryption function. For instance [33] presented a fully homomorphic encryption scheme that he claimed to be efficient for practical applications. He avoided bootstrapping and modulus switching as noise reduction approaches that were evident in previous FHE schemes but included it in the algorithm. According to [34], they proposed two fully homomorphic encryption schemes (symmetric and asymmetric) which were noiseless in the sense there was no *noise* factor contained in the ciphertexts.

Since the introduction of privacy homomorphism by [29], all schemes have been insecure until Gentry introduced the first fully homomorphic encryption scheme based on ideal lattices [35]. More improvements have been built on Gentry [7] with the simplest done by [32] that considered known attacks on the Approximate Greatest Common Divisor problem for two numbers (x_0, x_1) and many numbers (x_0,…….x_t). These attacks were mainly explored with a view of solving the approximate greatest common divisor problem, which is the secret key, p.

In the study of the attack on fully homomorphic encryption scheme, lattice attack based on the public key is considered. For instance in the security of FHE scheme, it was proved by [32] that it is equivalent to solve the approximate greatest common divisor problem. Consider a list of approximate multiples of *p*:

$$\{x_i = q_i p + r_i : q_i \in Z \cap [0, 2^{\gamma}/p),\ r_i \in Z \cap (-2^{\rho}, 2^{\rho})\}_{i=0}^{\tau} \quad \text{……..Eqn. 2}$$

According to FHE scheme, it is already known that an arbitrary ciphertext *c* has a general form: c = qp + 2r + m. In order to execute the attack here, it is very simple, that is, how to remove *qp* in a ciphertext *c* by adding small noise value. When completing this, it is easy to recover the plaintext *m* bit in *c*. The success of this calls for applying Diophantine Inequality Equation (DIE) problem [36].

The study has explored the noise term and its significant contribution to the performance of the FHE scheme especially where the noise factor impact heavily on the decryption function of the scheme. Thereafter the study has also explored the principle behind the attack in FHE scheme where the application of DIE problem has been proposed. Both the noise term and attacks impacts proportionally on the efficiency of the FHE schemes.

After the breakthrough in [7], the first attempted implementation of fully homomorphic encryption was the [37] implementation based Gentry's original cryptosystem where they reported timing of about thirty minutes per basic bit operation. In further evaluation of the same, an implementation from a variant of the [32] cryptosystem, reported evaluation of a complex circuit in thirty-six hours. While using the packed-ciphertext techniques, that implementation could evaluate the same circuit on fifty-four different inputs in the same thirty-six hours, yielding amortized time of roughly forty minutes per input. The AES-encryption circuit was then adopted as a benchmark in several subsequent works [38] thus improving the evaluation time to about four hours and the per-input amortized time to over seven seconds.

According to [32] in their study of efficiency of fully homomorphic encryption schemes, they improved on the previous works by introducing re-linearization and dimension-modulus reduction techniques. They argued that all somewhat homomorphic encryptions can be based on LWE by using the re-linearization technique since all previous schemes depended on the complexity assumptions based on ideals in various rings. They also deviated from the *squashing* approach used in previous works by introducing a new dimension-modulus reduction technique where the ciphertexts are shorten and thus reduces the decryption complexity of their scheme without bothering introducing any other assumptions.

III. METHODOLOGY, FINDINGS AND DISCUSSIONS

3.1 METHODOLOGY

In this paper, we focused on the time and space complexity of the algorithm, and the correctness of the solutions generated by the algorithm. Given that the objective was to develop an improved encryption scheme that enhances data security in cloud computing, an experimental research was adopted where combination of formal and theoretical

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

methodologies was found to be of much help. The newly developed scheme was coded and tested in a computing environment to ascertain that it is implementable. The data results generated was benchmarked with existing findings to prove its reliability and validity.

3.1.1 Test Lab Setup: The encryption algorithm of the new scheme was developed from the mathematical proofs realized. We considered both hardware and software of such test environment. On the hardware, it was tested on an AMD Quad-Core Processor 1.5 GHz with 4 GB DDR3 RAM Memory. The software component of the lab set up consisted of Windows 10 operating system (OS) and Java Runtime Environment for coding, compiling, debugging and test running of the code.

3.1.2 Metrics and Evaluation Parameters: The implementation of homomorphic encryption schemes considers several requirements and challenges. It is important to check the performance of the algorithm because of its impacts on the resource utilization. We evaluated the ACFHE Scheme using parameters: encryption time, decryption time, key generation time, and ciphertext size. This is in line with evaluation of an effective algorithm, that solves both time and space complexity.

3.1.3 Implementation Scenario: We focused on the use of integer values to demonstration the practicability of the new ACFHES algorithm. This was formed by the data representation in a classical computer, which operates on binary digits at the lowest definition.

3.1.4 Input Data Sets: With the scenario indicated above, we relied on integer data values to test the implementation of the ACFHE Scheme. This also supports the fact that the nature of data sets does not affect the performance of homomorphic encryption schemes but the size.

3.1.5 Benchmark Suite: The implementation of homomorphic encryption scheme can be benchmarked by other similar schemes that has been developed and implemented. ACFHES is asymmetrical probabilistic scheme that compares well with Gentry's construction, given that both are fully homomorphic encryption schemes.

3.2 ACFHES ALGORITHMN DEVELOPMENT

In the development of the ACFHES algorithm, the following steps were considered:

3.2.1 Designing the algorithm: We considered three main functions i.e. the key generation that relies on performing Euler Quotient function on prime numbers, the encryption function that concerns enciphering the input data sets, and the decryption function that decipher the ciphertext back into consumable output, plain data set.

3.2.2 Coding of the derived algorithm: This was done in Java programming language. Given that cloud serves are implemented web applications, Java as a programming language performs better. There were separate codes to capture the various component of the ACFHES scheme.

3.2.3 Benchmarking: It was compared against existing findings. An algorithm is required to meet the threshold of time and space complexity meaning the amount of computer processing power (CPU) and the amount of memory used in the run time.

3.2.4 Testing the code: The computer program generated from the developed algorithm was tested. The debugging process was considered to identify possible errors and appropriate correction done. The profiling or performance measurement was achieved by executing the program on data sets and measuring the time and space needed to realize results.

3.3 THE ACFHES ALGORITHM

This new scheme works in the polynomial ring $R = Z[x]/(f(x))$ with $f(x) = \Phi_d(x)$, the d-th cyclotomic polynomial of degree $n = \varphi(d)$. A plaintext is an element in the ring R_t for some modulus t where in this case t is taken as 2. A ciphertext in this scheme consists of elements in the ring R_q where q is the larger modulus. The security of this scheme is determined by the degree n of f, the size of q, and by the probability distributions. This ACFHE scheme uses an encryption scheme and two additional functions ADD and COMPO to perform function evaluations homomorphically on the encrypted data. The functions used in this scheme are summarized as below:

- ParamsGen (λ): for a given security parameter, choose a polynomial $\Phi_d(x)$, ciphertext modulus q and plaintext modulus t, and distribution key χ_{key}. Return the system parameters ($\Phi_d(x), q, t, \chi_{key}$) and set the plaintext modulus t=2.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

- KeyGen ($\Phi d(x)$, q, t, χ_{key}, w): Sample polynomial s from χ_{key}, sample $a \leftarrow R_q$, uniformly at random. Compute $b=[-(as)]_q$. The public key consists of two polynomials *pk ={b,a}* and the secret key sk=s.
- Encrypt (pk,m): the input message $m \in R_t$ is first encoded into polynomial $\Delta m \in R_t$ with $\Delta = \lfloor q/t \rfloor$. The ciphertext is the pair of polynomials $c = \{c_0, c_1\}$.
- Decrypt (sk, c): the polynomial is computed first $m = [c_0+sc_1]_q$ and then when $t = 2$, recover the plaintext message *m* by a decoding the coefficients of *m*. this decoding operation checks if the coefficients is in (*q/4, 3q/4*) for a 1 bit and 0 bit otherwise.
- Add (c_1, c_2): for two ciphertexts $c_0 = \{c_0,0,c_1,1\}$ and $c_1 = \{c1,0,c1,1\}$, return $c = \{c_0,0 + c_1,0, c_1,0 + c_1,1\}$.
- Compo (c_1, c_2): compute $c_{compo} = \{c_0,c_1,c_2\}$ where $c_0 = \lceil \lfloor t/q.c_1,0.c_2,0 \rfloor \rceil q$, $c_1 = \lceil \lfloor t/q.(c_1,0.c_2,1 + c_1,1.c_2,0) \rfloor \rceil q$, and $c_2 = \lceil \lfloor t/q.c_1,1.c_2,1 \rfloor \rceil q$.

In this scheme, only one bit is encrypted in one ciphertext and the encrypted bit remains in the least significant coefficient of the ciphertext polynomial. When it comes to decryption, only the coefficient of the least significant coefficient of m is decoded. The key generation algorithm generates the public and private keys that are to be used in the encryption process. This is necessary to enable achieve the key to be used in encryption and decryption process. The condition to be satisfied for the two chosen prime numbers is that both primes need to be of equivalent length, that is, $p,q \in 1||\{0,1\}^{s-1}$ with *s* being the security parameter. Usually there are two ways of selecting the *g* generator. The other way is to use the equation $g = (\alpha n + 1)\beta^n \bmod n^2$. The public key is (n,g) and the private key is (λ,μ). The encryption algorithm was the one used in the encipherment process, which provided the desired ciphertext *c*. The integer values *m* and *r* were provided and used in the calculation of ciphertext. The computed value was then displayed. The decryption algorithm above was used in the decipherment process, which reversed the ciphertext *c* to plaintext *m*. The calculated plaintext *m* was displayed to confirm the original text before the encryption process. In Addition algorithm, the homomorphic property of the algorithm is tested. The entered plaintext value is summed to prove the homomorphic property when compared. In Composition algorithm, the composition property is tested by computing the composition of the two values and this proves the homomorphic property of the algorithm. The values entered produce the composed results thus makes the plaintext "invisible" to anybody who access it.

3.4 IMPLEMENTATION AND TESTING OF ACFHES

The AFCHES was benchmarked with other homomorphic encryption schemes for purposes of evaluating its efficiency. Efficiency is relative and not absolute, and can be explained under two main guidelines: time complexity and space complexity. The time complexity explains the time taken to complete the encryption and decryption process. The space complexity define the consumed computing storage resources at the encryption and decryption time. It is about the number of memory cells needed to carry out computational steps required to solve an instance of a problem excluding the space allocated to hold the inputs [39]. In this study, we adopted the simulational method since other homomorphic encryption schemes have been implemented and efficiency results are available therefore the remaining bit is to run results for ACFHES and compare them against existing ones. Moreover the computational power of implementation environment adopted here provided enough ground for effective comparison given that it was done under minimal specifications. The measurement of both encryption and decryption time can be achieved through the use of java application using smart card I/O API available in Java SE 6 and the System.nanoTime() method included in the API, which returns the current time in nanoseconds of the most precise available system time. The nanoTime which measures elapsed time and depends on the underlying architecture is significantly more accurate than currentTime Millis that measures wall-clock time but it is an expensive call as well. Therefore System.nanoTime() guarantees a nanosecond precise time relative to arbitrary point. The developed code when run generated results as shown in table below. Each algorithm were timed differently to ascertain the computational time elapsed in each case. The data results were obtained after carrying out 20 times with an elapse time difference of 1 minute. This was done to enable "refreshing" the computational activity that may have been ignited by the previous operation. A mean value was computed to arrive at the average operation timing as captured in the table 4 below.

Table 4 Operation Timings in Milliseconds

Operation	**λ = 512**	**λ = 1024**	**λ = 2048**
KeyGen	15.83	93.07	2056.19
Encrypt	15.81	133.31	1453.77
Decrypt	1.45	1.87	2.36
Add	1.46	1.99	2.44
Compo	1.49	2.35	2.82

From the data displayed in Table 4 above, the algorithms for addition, composition and decryption are atomic operations for integer handling, and the most time consuming operations are the KeyGen and Encrypt since the

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

determination of a prime number takes relatively long runtime period. It took fairly longer to generate keys and encrypt the input data as compared to decryption time and time for testing the homomorphic properties: addition and composition.

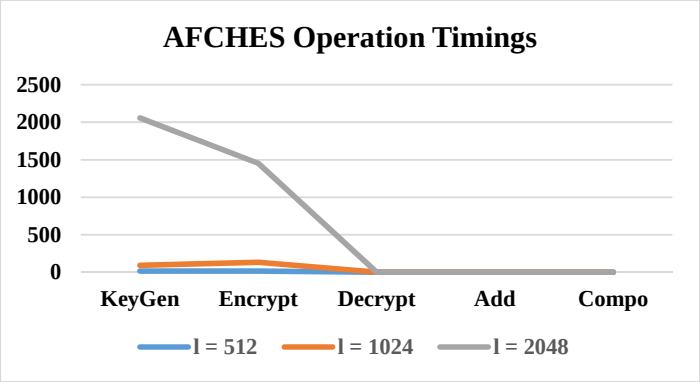


Fig. 7 AFCHES Operation Timings

Fig. 7 above was a graphical interpretation of the data captured in table 4 based on the considered basic operations of the scheme. The same data results were compared with a similar experiment done under Java 6 SE on a 2.4 GHz Core 2 Duo with 3 MB L2 cache and 4 GB RAM [40]. Their results were as also in milliseconds and were as follows:

Table 5 Timings for Basic Operations [55]

Operation	λ = 512	λ = 1024	λ = 2048
KeyGen	35	270	4242
Encrypt	35	301	3218
Decrypt	1	1	1
Add	1	1	1
Mult	1	1	1

The analysis of the same data from [40] was also graphical presented as in Fig. 8 below. The two graphs shows that the values obtained from AFCHES were better in terms of efficiency as measured compared to the earlier one done by [40].

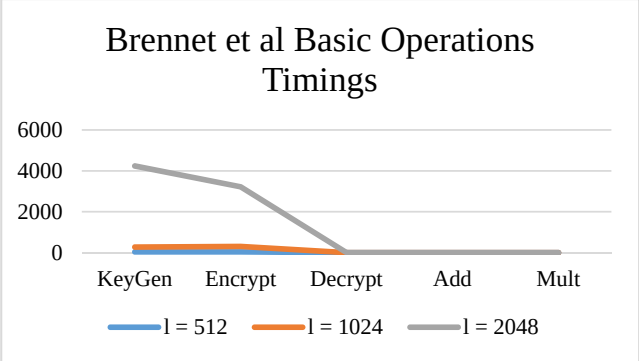


Fig 8 Basic Operations Timings [55]

Partial homomorphic operations are found to restrict the range of circuits that they support since they depend on one mathematical operations whereas fully homomorphic encryption schemes evaluate any Boolean circuits but computation speed and ciphertext size found to affects their efficiency. There are requirements that supports the effective evaluation of homomorphic schemes, that is, versality, speed, and ciphertext size in an encryption scheme.

Table 6 Homomorphic Encryption Schemes Comparison

Type	*Versatility*	*Speed*	*Ciphertext Size*
PHE	Low	Fast	Small
FHE	High	Slow	Large
AFCHES	High	Faster	Medium

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

IV CONCLUSIONS AND FUTURE WORK

The study has reviewed both cloud computing and homomorphic encryption schemes, and shown how significant their relationship advantage data security. We have also appreciated previous contributions that has been made to improve security of data while in cloud computing. Our findings still agree with the fact that there is need for a better candidate for improving security of data in cloud computing and formed our main motivation. The main focus was to demonstrate how suitable the newly developed Addition-Composition Fully Homomorphic Scheme for securing data within cloud computing. The new scheme has posted impressive results, which when compared with previous ones, performs much better. It is also important to note that our testbed used a less superior computing environment thus point to high possibility of better results when both are tested on same environment. For future work, the study recommends that the performance of our scheme can be tested further on computing environment with superior specifications to enable the registration of better results.

REFERENCES

[1] Rittinghouse John and Ransom James, “Cloud Computing: Implementation, Management and Security”, CRC Press, Taylor & Francis Group, 2010, pp 340.

[2] Kant C. and Sharma Y, “Enhanced Security Architecture for Cloud Data Security”. International Journal of Advanced Research in Computer Science and Software Engineering. Vol. 3 Issue 5 ISSN: 2277128X, 2013, pp. 570 – 575.

[3] Vaquero L.M, Rodero-Merino L., Caceres J. and Lindner M. “A break in the clouds: towards a cloud definition”, in: ACM SIGCOMM Computer Communication Review, 2008, p.50-55.

[4] Mollah B. M., Islam K.R., and Islam S.S. “Next generation of computing through cloud computing technology”, in: 2012 25th IEEE Canadian Conference on Electrical Computer Engineering (CCECE), May 2012, p.1-6.

[5] Ukil, A., Jaydip S., Bera, D., and Pal, A. “A Distributed Intrusion Detection System for Wireless Ad Hoc Networks”. In *Proceedings of the 16th IEEE International Conference on Networking (ICON’08)*, pp. 1-5, 2008, New Delhi, India.

[6] Agudo I, Nunez D., Giammatteo G., Rizomiliotis P., and Lambrinoudakis C. “Cryptography goes to the Cloud. Secure and Trust Computing, Data management, and Applications”, Vol. 187, 2011 pp 190 – 197. Springer Berlin Heidelberg.

[7] Gentry Craig, “A Fully Homomorphic Encryption Scheme” PhD Thesis in Computer Science, Department of Computer Science, Stanford University, California, USA. 2009

[8] Weinman J. “Axiomatic Cloud Theory”. Communications, Media and Entertainment Industry for Hewlett-Packard, 2011.

[9] Pranita P. K. and Ubale V. S. “Cloud Computing Security Issues and Challenges” International Refereed Journal of Engineering and Science (IRJES). 2013. Vol. 2 Issue 2 pp 10 – 13.

[10] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinsky, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M (2009). “Above the Clouds: A Berkley View of Cloud Computing”. Technical Report No. UCB/EECS-2009-28, Department of Electrical Engineering and Computer Sciences, University of California at Berkley. February 10, 2009. Available on line at: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf (Accessed on: November 2016).

[11] Cloud Security Alliance (CSA)’s “Security Guidance for Critical Areas of Focus in Cloud Computing”. CSA, April 2009. Available Online at: https://cloudsecurityalliance.org/csaguide.pdf (Accessed on: November 2017).

[12] Navneet S. P. and Rekha B. S. “Software as a Service (SaaS): Security issues and Solutions”. International Journal of Computational Engineering Research (IJCER). ISSN (e): 2250 – 3005, Vol. 04, Issue, 6, June – 2014.

[13] Seccombe A., Hutton A., Meisel A., Windel A., Mohammed A. and Licciardi A. “Security guidance for critical areas of focus in cloud computing”, v2.1. Cloud Security Alliance, 2009, 25 p. pp 3121–3124.

[14] Tiwari P. K. and Mishra B. “Cloud Computing Security Issues, Challenges and Solution”. International Journal of Emerging Technology and Advanced Engineering. Vol. 2 Issue 8 ISSN: 2250-2459, 2012.

[15] Mehmet T. S. and Harmanci A. E. “Security Problems of Platform-as-a-Service (PaaS) Clouds and Practical Solutions to the Problems”. 2012 31st International Symposium on Reliable Distributed Systems, IEEE Computer Society, pp 463 – 468.

[16] Devi T. and Ganesan R. “Platform-as-a-Service (PaaS): Model and Security Issues”. TELKOMNIKA Indonesian Journal of Electrical Engineering, Vol. 15, No. 1, July 2015, pp. 151 ~ 161.

[17] Gnanavelu D. and Gunasekaran G. “Survey on Security Issues and Solutions in Cloud Computing”. International Journal of Computer Trends and Technology. 2014. Vol. 8 No. 3. Pp 126 – 130. ISSN: 2231-2803.

[18] Jaydip S. and Sengupta, I. “Autonomous Agent-Based Distributed Fault-Tolerant Intrusion Detection System”. In *Proceedings of the 2nd International Conference on Distributed Computing and Internet Technology (ICDCIT’05)*, pp. 125-131, December, 2005, Bhubaneswar, India. Springer LNCS Vol. 3186.

[19] Jaydip S., Sengupta I., and Chowdhury P. R. “An Architecture of a Distributed Intrusion Detection System Using Cooperating Agents”. In *Proceedings of the International Conference on Computing and Informatics (ICOCI’06)*, pp. 1-6, June, 2006, Kuala Lumpur, Malaysia.

[20] Ukil A., Jaydip S., Bera D., and Pal A. “A Distributed Intrusion Detection System for Wireless Ad Hoc Networks”. In *Proceedings of the 16th IEEE International Conference on Networking (ICON’08)*, pp. 1-5, December 2005, New Delhi, India.

[21] Jaydip S. “An Agent-Based Intrusion Detection System for Local Area Networks”. *International Journal of Communication Networks and Information Security (IJCNIS)*, Vol 2, No 2, pp. 128-140, August 2010.

[22] Jaydip S. “A Robust and Fault-Tolerant Distributed Intrusion Detection System”. In *Proceedings of the 1st International Conference on Parallel, Distributed and Grid Computing (PDGC’10)*, pp. 123-128, October 2010, Waknaghat, India.

[23] Trusted Computing Group (TCG)’s White Paper. “Cloud Computing and Security- A Natural Match”. Available online at: http://www.trustedcomputinggroup.org (Accessed on; November 2017).

[24] Jaydip S. “Security and Privacy Issues in Cloud Computing”. Innovation Labs, Tata Consultancy Services Ltd, Kolkata, India. 2015.

[25] Kumar K. S., Anjaneyulu N. and Venkanna. “Security Issues in Cloud Computing and study on Encryption Method”. International Journal of Emerging Trends and Technology in Computer Science. 2013. Vol. 2 Issues 3, pp 442 – 446, ISSN: 2278 – 6856.

[26] Shucheng Y., Cong W., Kui R. and Wenjing L. “Achieving Secure, Scalable and fine-grained data access control in cloud computing”, in: IN-FOCOM, 2010 Proceedings IEEE, 2010.p.1-9.

[27] Velumadhava R. R. and Selvamani K. “Data Security Challenges and Its Solutions in Cloud Computing”. International Conference on Intelligent Computing, Communication & Convergence (ICCC-2015). Procedia Computer Science 48, pp 204 – 209.

[28] Chakraborty N. “Cloud Security using Homomorphic Encryption”. National Conference on Advances in Computing, Networking and Security, 2013.

[29] Rivest R. L., Adleman L. and Dertouzos M. L. “On Data Banks and Privacy Homomorphisms”, chapter on Data Banks and Privacy Homomorphisms, 1978, pp. 169 – 180, Academic Press.

[30] Boneh D., Goh E. and Nissim K. “Evaluating 2-DNF Formulas on Ciphertexts”. In Theory of Cryptography Conference. 2005.

[31] Lopez –Alt A., Tromer E., and Vaikuntanathan V. “On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption”. In STOC 2012 (ACM).

[32] Djik M., Gentry C., Halevi S., and Vaikuntanathan V. “Fully homomorphic encryption over the integers”. In Proc. of Eurocrypt, Vol. 6110 of LNCS, 2010, pp 24-43. Springer.

[33] Liu. “Practical Fully Homomorphic Encryption without Noise Reduction”. IACR Cryptology ePrint Archive, vol. 468, 2015.

[34] Jing Li and Licheng Wang. “Noiseless Fully Homomorphic Encryption”. 2017. Cryptology ePrint Archive.

[35] Chungseng. “Attack on Fully Homomorphic Encryption Scheme over Integers”. Cryptography and Security, 2012, pp 24.

[36] Chungseng. “Attack on Fully Homomorphic Encryption Scheme over Integers”. Cryptography and Security, 2012, pp 24.

[37] Gentry C. and Halevi S. “Implementing Gentry’s Fully-Homomorphic Encryption Scheme”. EUROCRYPT’11 Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology. ISBN: 978-3-642-20464-7. Springer-Verlog Berlin, Heidelberg. 2010.

[38] Dai W., Doroz Y, and Sunar B “Accelerating NTRU based Homomorphic Encryption using GPUs”. Proceedings on 2014 IEEE High Performance Extreme Computing Conference, pp 1 – 6, ISBN: 978-1-4799-6232-7.

[39] Horowitz E., Sahni S., and Saguthevar R. “Fundamentals of Computer Algorithms”. Galgotia Publications pvt Ltd, India.1998.

[40] Brenner M, Wiebelitz J, Voigt G, and Smith M. “Secret Program Execution in the Cloud Applying Homomorphic Encryption”. 5th IEEE International Conference on Digital Ecosystems and Technologies, Daejeon, Korea. 2011. pp 114 – 119. ISBN: 978-1-4577-0872-5.